Neural Computation **11(1)**, 193–213, 1999.

Variational Learning in Nonlinear Gaussian Belief Networks

Brendan J. Frey and Geoffrey E. Hinton
Department of Computer Science, University of Toronto
6 King's College Road, Toronto, Canada M5S 3G4

October 10, 1998

Abstract

We view perceptual tasks such as vision and speech recognition as inference problems where the goal is to estimate the posterior distribution over latent variables (*e.g.*, depth in stereo vision) given the sensory input. The recent flurry of research in independent component analysis exemplifies the importance of inferring the continuous-valued latent variables of input data. The latent variables found by this method are linearly related to the input, but perception requires nonlinear inferences such as classification and depth estimation. In this paper, we present a unifying framework for stochastic neural networks with nonlinear latent variables. Nonlinear units are obtained by passing the outputs of linear Gaussian units through various nonlinearities. We present a general variational method that maximizes a lower bound on the likelihood of a training set and give results on two visual feature extraction problems. We also show how the variational method can be used for pattern classification and compare the performance of these nonlinear networks with other methods on the problem of handwritten digit recognition.

1 Introduction

There have been many proposals for unsupervised, multilayer neural networks that contain a stochastic generative model and learn by adjusting their parameters to maximize the likelihood of generating the observed data. Two of the most tractable models of this kind are factor analysis (Everitt 1984) and independent component analysis (Comon, Jutten and Herault 1991; Bell and Sejnowski 1995; Amari, Cichocki and Yang 1996; MacKay 1997).

1.1 Linear generative models

In factor analysis there is one hidden layer that contains fewer units than the visible layer. In the generative model, the hidden units are driven by zero-mean, unit-variance, independent Gaussian noise. The hidden units provide top-down input to the linear visible units via the generative weights and each visible unit has its own level of added Gaussian noise. Given the generative weights and the noise levels of the visible units, it is tractable to compute the posterior distribution of the hidden activities that is induced by an observed vector of visible activities. This posterior distribution is a full covariance Gaussian whose mean depends on the visible activities. Once this distribution has been computed it is straightforward to adjust the generative weights to maximize the likelihood of the observed data using either a gradient method or the expectation maximization (EM) algorithm (Rubin and Thayer 1982). Unfortunately, factor analysis ignores all the statistical structure in the data that is not contained in the covariance matrix and its hidden representations are linearly related to the data, so it is unable to extract many of the hidden causes of the data that are important in tasks such as vision and speech recognition.

In independent component analysis the generative model is still linear, but the independent noise levels for the hidden units are non-Gaussian. This makes it difficult to compute the full posterior distribution across the hidden units given a visible vector. However, by using the same number of hidden and visible units and by setting the noise levels of the visible units to zero, it is possible to collapse the posterior distribution across the hidden units to a point which is found by multiplying the visible activities by the inverse of the matrix of hidden-to-visible generative weights. To maximize the likelihood of the data, the weights are adjusted to make the posterior points have high log probability under the noise models of the hidden units, whilst also keeping the determinant of the generative weight matrix small so that probability density in the space of hidden activities gets concentrated when it is mapped into the visible space. Unfortunately, independent component analysis extracts components that are a linear function of the data and it assumes the data is noise-free, so it too is unable to extract hidden causes that are nonlinearly related to observed, noisy data. Recently, attempts have been made to enhance the representational capabilities of independent component analysis by adding noise to the visible units (Olshausen and Field 1996; Lewicki and Sejnowski 1998).

1.2 Very nonlinear generative models

An appealing approach to understanding how the cortex constructs models of sensory data is to assume that it uses maximum likelihood to learn a hierarchical generative model. For tasks such as vision and speech recognition, the cortex probably requires distributed representations that are a nonlinear function of the data and that allow noise at every level of the hierarchy. Attempts at developing learning algorithms capable of constructing such generative models have been less successful in practice than the simpler linear models. This is because it is hard to compute (or even to represent) the posterior probability distribution across the hidden representations when given a visible vector and a set of weights and noise variances.

The unsupervised version of the Boltzmann machine (Hinton and Sejnowski 1986) is a multilayer generative model which learns distributed representations that are a nonlinear function of the data. It uses symmetrically connected stochastic binary units and has a relatively simple learning rule which follows the gradient of the log likelihood of the data under the generative model. Unfortunately, to get this gradient it is necessary to perform Gibbs sampling in the hidden activities until they reach thermal equilibrium with a data vector clamped on the visible units. This is very time consuming and the problem is made even worse by the need to compute derivatives of the partition function which requires the network to reach thermal equilibrium with the visible units unclamped. The sampling noise and the difficulty in reaching equilibrium in networks with large weights make the learning algorithm painfully slow.

When binary stochastic units are connected in a directed acyclic graph we get a "binary sigmoidal belief network" (Pearl 1988; Neal 1992). (Here, "acyclic" means that there aren't any closed paths when following edge directions. There may be closed paths when the edge directions are ignored.) The net input to each unit is given by a weighted sum of the activities of the unit's parents. Learning is easier in this network than in a Boltzmann machine because there is no need to compute the derivative of a partition function and the gradient of the log likelihood does not involve a difference in sampled statistics. Most importantly, it is no longer necessary for the Gibbs sampling to converge to thermal equilibrium before the weights are adjusted. Using the analysis of EM provided by Neal and Hinton (1993), it can be shown that on average the learning algorithm improves a bound on the log probability of the data even when the Gibbs sampling is too brief to get close to equilibrium (Hinton, Sallans and Ghahramani 1998).

There have been several attempts to avoid Gibbs sampling altogether when fitting a sigmoidal belief network to data. (See (Frey 1998) for a review of these methods.) They all rely on the idea that learning can still improve a bound on the log likelihood of the data even when the posterior distribution over hidden states is computed incorrectly. The stochastic Helmholtz machine (Hinton et al. 1995) uses a separate, stochastic recognition network to compute a quick and dirty approximation to a sample from the posterior distribution over the hidden units when given a visible vector. There is a very simple rule for learning both the generative weights and the recognition weights, but the approximation produced by the recognition network is often poor and the method of learning the recognition weights is not guaranteed to improve it. The deterministic Helmholtz Machine (Dayan et al. 1995) makes even more restrictive assumptions than the stochastic version about the probability distribution that is used to approximate the full posterior distribution over the binary hidden states when given a data vector. It assumes that the approximating distribution can be written as a product of separate probabilities for each hidden unit. It also assumes that the approximating product distribution can be computed by a deterministic recognition network in a single bottom-up pass. This latter assumption is relaxed in variational approaches (Saul, Jaakkola and Jordan 1996; Jaakkola, Saul and Jordan 1996) which eliminate the separate recognition model and use the generative weights and numerical optimization to find the set of probabilities that minimizes the asymmetric divergence from the true posterior distribution.

1.3 Continuous sigmoidal belief networks

For real-valued data that comes from real physical processes, binary units are often an inappropriate model because they fail to capture the approximately linear structure of the data over small ranges. For example, very small changes in the position, orientation, or scale of an object lead to linear changes in the pixel intensities. One way to endow the linear Gaussian networks described above with representations that are nonlinear functions of the data is to apply a smooth sigmoidal squashing function to the output of each Gaussian before passing the activity down the network. The nonlinear squashing function allows each unit to take on a variety of behaviors, ranging from nearly Gaussian to nearly binary. In (Frey 1997a) and (Frey 1997b), it was shown that Markov chain Monte Carlo and a variational method could be used to train small networks of these units. However, the smoothness of the squashing function prevents units from placing probability mass on a single point, and so these units are not able to produce activities exactly equal to zero. The ability of a network to set activities exactly equal to zero is important for sparse representations where many units do not participate in explaining an input pattern.

1.4 Piecewise linear belief networks

In an attempt to produce sparse distributed representations of real-valued data, Hinton and Ghahramani (1997) investigated generative models composed of multiple layers of rectified linear units. In the generative model, each unit receives top-down input that is a linear function of the rectified states in the layer above and it adds Gaussian noise to get its own real-valued unrectified state,

$$x_i = w_{i0} + \sum_{j \in A_i} w_{ij} f(x_j) + \text{noise}, \quad \text{where} \quad f(x_j) = \begin{cases} 0 & \text{if } x_j < 0, \\ x_j & \text{if } x_j \geq 0, \end{cases} \tag{1}$$

and A_i is the set of indices for the parents of unit i. The output that a unit sends to the layer below is equal to its unrectified state if it is positive but is equal to 0 if it is negative.

Networks of these units can set the activities of some units exactly equal to zero so that they do not participate in explaining the current input pattern. Hinton and Ghahramani (1997) showed that Gibbs sampling was feasible in such networks and that multilayer networks of rectified linear units could learn to extract sparse hidden representations that were nonlinearly related to images.

1.5 Nonlinear Gaussian belief networks

Linear generative models, binary sigmoidal belief networks, continuous sigmoidal belief networks, and piecewise linear belief networks can all be viewed as networks of Gaussian units that apply various nonlinearities to their Gaussian states. The probability density function over the pre-nonlinearity variables $\mathbf{x} = (x_1, \ldots, x_N)$ in such a nonlinear Gaussian belief network (NLGBN) is

$$p(\mathbf{x}) = \prod_{i=1}^{N} p(x_i | \{x_j\}_{j \in A_i}) = \prod_{i=1}^{N} \frac{1}{\psi_i} \phi\Big(\frac{x_i - \sum_{j \in A_i} w_{ij} f_j(x_j)}{\psi_i} \Big), \tag{2}$$

where A_i is the set of indices for the parents of unit i and $\phi(\cdot)$ is the standard normal density function:

$$\phi(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2}. \tag{3}$$

ψ_i^2 is the variance of the Gaussian noise for unit i and $f_j(\cdot)$ is the nonlinear function for unit j. For example, some units may use a step function (making them binary sigmoidal units with a cumulative Gaussian activation function), whereas other units may use the rectification function (making them real-valued units that encourage sparse representations). We define $f_0(x_0) = 1$ so that w_{i0} represents a constant bias for unit i in (2).

In this paper we generalize the variational method developed by Jaakkola *et al.* (1996) for networks of binary units and show that it can be successfully applied to performing approximate inference and learning in nonlinear Gaussian belief networks. The variational method can still be applied when different types of nonlinearity are used in the same network, such as networks of the kind described in (1998), where binary and linear units come in pairs and the output of each linear unit is gated by its associated binary unit.

2 Variational Expectation Maximization

A surprisingly simple variational technique can be used for inference and learning in NLGBNs. In this method, once some variables have been observed, we postulate a simple parametric *variational distribution* $q(\cdot)$ over the remaining unobserved variables. (The variational distribution $q(\cdot)$ is separate from the generative distribution $p(\cdot)$.) A numerical optimization method (*e.g.*, conjugate gradients) is then used to adjust the variational parameters to bring $q(\cdot)$ as "close" to the true posterior as possible. We use a cost function that not only measures "closeness" in the Kullback-Leibler sense, but also bounds from below the log-likelihood of the input pattern. This choice of cost function leads to an efficient generalized expectation maximization learning algorithm, as described below.

2.1 A cost function that bounds the log-probability

Let V be the set of indices of the observed variables for the current input pattern, and let H be the set of indices of the unobserved variables for the current input pattern, so that $V \cup H = \{1, \ldots, N\}$. The variational bound (Neal and Hinton 1993) is

$$F = \langle \log p(\mathbf{x}) \rangle - \langle \log q(\{x_i\}_{i \in H}) \rangle \leq \log p(\{x_i\}_{i \in V}), \tag{4}$$

where $\langle \cdot \rangle$ indicates an expectation over the unobserved variables with respect to $q(\cdot)$. It is easily shown that for unconstrained $q(\cdot)$, F is maximized by setting $q(\{x_i\}_{i \in H}) = p(\{x_i\}_{i \in H} | \{x_i\}_{i \in V})$ in which case the bound in (4) is tight. This gives exact probabilistic inference, whereas using a constrained form for $q(\cdot)$ gives approximate probabilistic inference.

The variational distribution we consider here is a product of Gaussians:

$$q(\{x_i\}_{i \in H}) = \prod_{i \in H} q(x_i) = \prod_{i \in H} \frac{1}{\sigma_i} \phi\Big(\frac{x_i - \mu_i}{\sigma_i}\Big), \tag{5}$$

where μ_i and σ_i, $i \in H$ are the variational parameters. By adjusting these parameters, we can obtain an axis-aligned Gaussian approximation to the true posterior distribution over the pre-nonlinearity hidden variables.

For this variational distribution, it turns out that F can be expressed in terms of the mean and variance of the *post-nonlinearity* activities. Let $M_i(\mu, \sigma)$ be the mean output of unit i when the input is Gaussian noise with mean μ and variance σ^2:

$$M_i(\mu, \sigma) = \int_x \frac{1}{\sigma} \phi\Big(\frac{x - \mu}{\sigma}\Big) f_i(x) dx. \tag{6}$$

Let $V_i(\mu, \sigma)$ be the variance at the output of unit i when the input is Gaussian noise with mean μ and variance σ^2:

$$V_i(\mu, \sigma) = \int_x \frac{1}{\sigma} \phi\Big(\frac{x - \mu}{\sigma}\Big) \big\{ f_i(x) - M_i(\mu, \sigma) \big\}^2 dx. \tag{7}$$

We assume that these can be easily computed, closely approximated, or in the case of $V_i(\cdot,\cdot)$, bounded from above (the latter will give a new lower bound on F). See App. C for these functions in the case of linear units, binary units, rectified units, and sigmoidal units.

The variational bound (4) simplifies to[1]

$$F = -\sum_{i=1}^{N} \frac{1}{2\psi_i^2}\left\{ \left[\mu_i - \textstyle\sum_{j\in A_i} w_{ij} M_j(\mu_j,\sigma_j)\right]^2 + \textstyle\sum_{j\in A_i} w_{ij}^2 V_j(\mu_j,\sigma_j) \right\}$$
$$+ \sum_{i\in H} \frac{1}{2}\left(1 + \log 2\pi\sigma_i^2 - \frac{\sigma_i^2}{\psi_i^2}\right) - \sum_{i=1}^{N} \frac{1}{2}\log 2\pi\psi_i^2. \tag{8}$$

To make this formula concise, we have introduced dummy variational parameters for the observed variables: if x_i is observed to have the value x_i^*, we *fix* $\mu_i = x_i^*$ and $\sigma_i = 0$.

For unit i in (8), the term in curly braces measures the mean squared error under $q(\cdot)$ between μ_i and the input to unit i as given by its parents: $\langle[\mu_i - \sum_{j\in A_i} w_{ij} f_j(x_j)]^2\rangle$. It is down-weighted by the model noise variance ψ_i^2, since for larger noise variances, a particular mean squared prediction error is less important.

2.2 Probabilistic inference

Variational inference consists of first fixing $\mu_i = x_i^*$ and $\sigma_i = 0$, $i \in V$ in (8) and then maximizing F with respect to μ_i and $\log\sigma_i^2$, $i \in H$. (The optimization for the variances is performed in the $\log$-domain, since $\log\sigma_i^2$ is allowed to go negative.) We use the conjugate gradient method to perform this optimization, although other techniques can be used (*e.g.*, steepest descent or possibly a covariant method (Amari 1985)). The derivatives of F with respect to μ_i and $\log\sigma_i^2$, $i \in H$ are given in App. A. After optimization, the means and variances of the variational distribution represent the inference statistics.

2.3 Learning

We bound the log-probability of an entire training set by $\mathcal{F}$, which is equal to the sum of the bounds for the individual training patterns. The variational expectation maximization algorithm based on $\mathcal{F}$ consists of iterating the following two steps:

---

[1]To see how $\langle\log p(\mathbf{x})\rangle$ simplifies, add and subtract both μ_i and $\sum_{j\in A_i} w_{ij} M_j(\mu_j,\sigma_j)$ in the numerator of the argument of $\phi(\cdot)$ in (2), so that $\langle\log p(\mathbf{x})\rangle = -\sum_i \log(2\pi\psi_i^2)/2 - \sum_i \langle([x_i - \mu_i] + [\mu_i - \sum_{j\in A_i} w_{ij} M_j(\mu_j,\sigma_j)] + \sum_{j\in A_i} w_{ij}[M_j(\mu_j,\sigma_j) - f_j(x_j)])^2\rangle/2\psi_i^2$. The cross-terms produced by the square vanish under the expectation.

- E-Step: Perform variational inference, by maximizing $\mathcal{F}$ with respect to the sets of variational parameters corresponding to the different input patterns.
- M-Step: Maximize $\mathcal{F}$ with respect to the model parameters ($w_{..}$'s and $\psi_{.}$'s).

Notice that by maintaining sufficient statistics while scanning through the training set in the E-Step, it is not necessary to store the sets of variational parameters. These sufficient statistics are described in App. B. However, to speed up the current E-Step, we initialize the set of variational parameters to the set found at the end of the last E-Step for the same pattern.

It turns out that the M-Step can be performed very efficiently (see App. B for details). Since the values of the model variances do not affect the values of the weights that maximize F in (8), we first maximize F with respect to the weights. As pointed out by Jaakkola *et al.* for their binary Gaussian belief networks, F is quadratic in the weights, so we can use singular value decomposition to solve for the weights exactly. Next, the optimal model variances are computed directly.

2.4 Software

A set of UNIX programs that implement variational learning in NLGBNs is available at `http://www.cs.utoronto.ca/~frey`. The software includes linear units, binary units, rectified units and sigmoidal units. New types of unit can be added easily by providing the nonlinear function and its derivatives.

3 Visual Feature Extraction

Approximate maximum likelihood estimation in latent variable models can be used to learn latent structure that is perceptually significant (Hinton et al. 1995). In this section, we consider two unsupervised feature extraction tasks and for each task we compare the representations learned by the variational method applied to two types of NLGBN and the representations learned by Gibbs sampling applied to a piece-wise linear NLGBN. If the hidden units all use a piece-wise linear activation function, then Gibbs sampling can be efficiently used for learning, as described in (Hinton and Ghahramani 1997). One of the NLGBNs used for variational learning contains only binary hidden units of the type described in (Jaakkola, Saul and Jordan 1996). So, in this section we see how the variational technique compares to another learning method for continuous hidden units, as well as how the generalization of the variational method from binary to continuous units compares to variational learning in binary networks.

3.1 The continuous bars problem

An important problem in vision is modeling surface edges in a way that is consistent with physical constraints. The goal of the much simpler "bars problem" (Dayan and Zemel 1995) is to learn without supervision to detect bars of two orthogonal orientations and to model the constraint that each image consists of bars of the same orientation. In (Hinton and Ghahramani 1997), a continuous form of this problem was presented. Each training image is formed by first choosing between vertical and horizontal orientation with equal probability. Then, each bar of that orientation is turned on with a probability of 0.3 with an intensity that is drawn uniformly from $[0, 5]$. Eight examples from a training set of 1000 6×6 images of this sort are shown in Fig. 1a, where the area of each tiny white square indicates the pixel intensity. A noisy version of this data in which unit-variance noise is added to each pixel is shown in Fig. 1e (a black square indicates a negative pixel value).

For each of the two data sets we used 100 iterations of variational EM to train a three-layer NLGBN with 1 binary top-layer unit, 16 rectified middle-layer units and 36 linear visible bottom-layer units. (Using more units in the hidden layers had little effect on the features extracted during learning.) The resulting weights projecting from each of the 16 middle-layer units to the 6×6 image are shown in Fig. 1b and Fig. 1f. Surprisingly, clearer bar features were extracted from the noisy data. The weights (not shown) from the top-layer binary unit to the middle-layer units tend to make the top-layer unit active for one orientation and inactive for the other. The weights look similar if a rectified unit is used at the top, but a binary unit properly represents the discrete choice between horizontal and vertical. Fig. 1c and Fig. 1g show the weights learned by variational EM in a network where all of the hidden units are binary. The individual bars are not properly extracted for either the noise-free or noisy training data. The log-probability bounds for the trained binary-rectified-linear NLGBNs are 27.4 nats and -60.3 nats, whereas the bounds for the trained binary-binary-linear NLGBNs are -48.3 nats and -65.6 nats.

We also trained a three-layer NLGBN with rectified hidden units using Gibbs sampling. For this method, a learning rate must be chosen and we used 0.1. 16 sweeps of Gibbs sampling were performed for each pattern presentation before the parameters were adjusted on-line. The weights obtained after 10 passes through each data set are shown in Fig. 1d and Fig. 1h. The features for the noise-free data are not as clear as the ones extracted using the variational method. The features for the noisy data can be cleaned up if some weight decay is used (Hinton and Ghahramani 1997). We did not estimate the log-probability of the data in this case, since Gibbs sampling does not readily provide a straight-forward way to obtain such estimates. In our experiments, variational EM and the Gibbs sampling method took roughly the same time. However, an on-line version of variational EM may be faster.

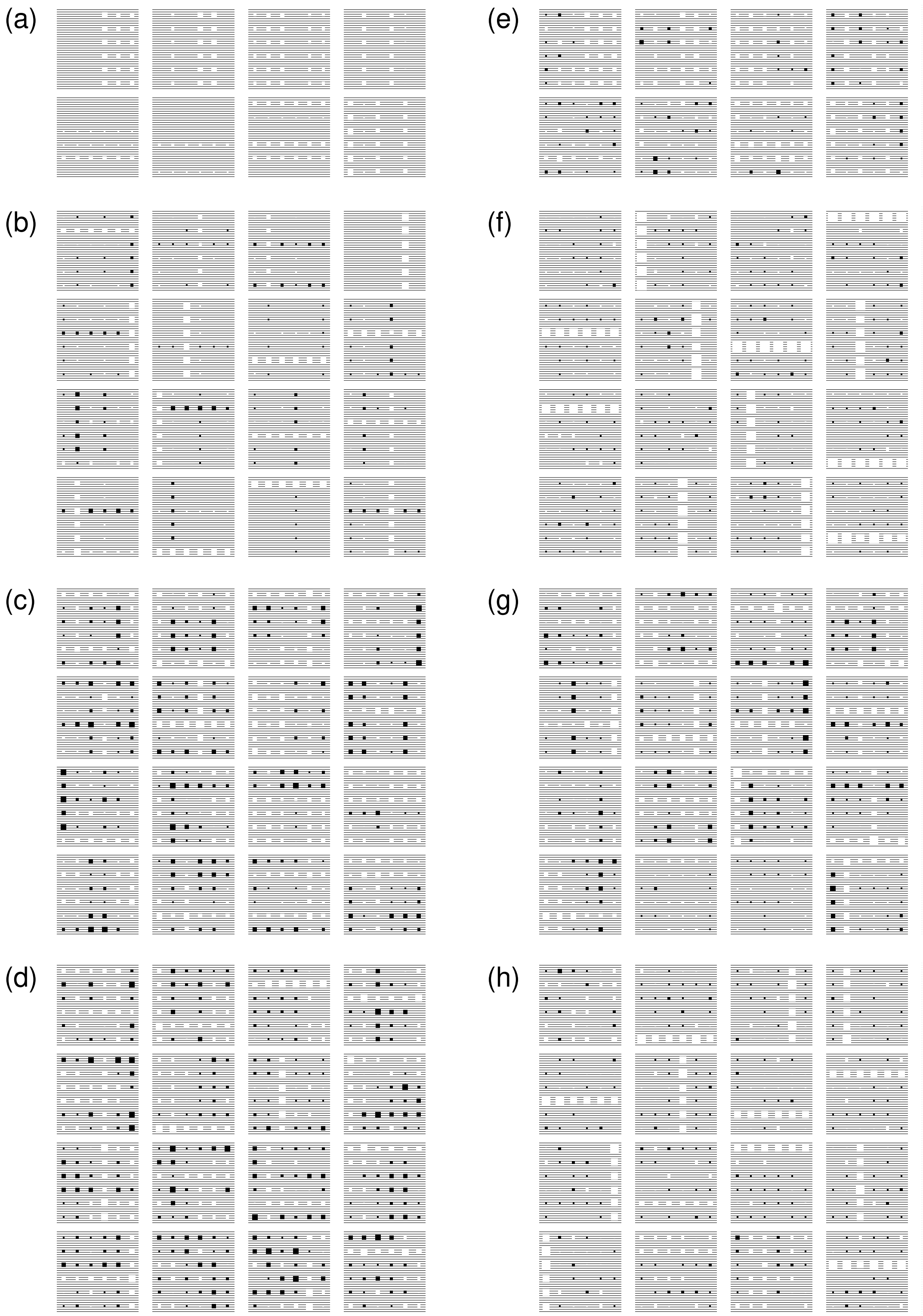

Figure 1: Learning in NLGBNs using the variational method and Gibbs sampling, for noise-free (a) - (d) and noisy (e) - (h) bar patterns. (a) and (e): Training examples. (b) and (f): Weights learned by the variational method with rectified units. (c) and (g): Weights learned by the variational method with binary units. (d) and (h): Weights learned by Gibbs sampling with rectified units.

3.2 The continuous stereo disparity problem

Another vision problem where the latent variables are nonlinearly related to the input is the estimation of depth from a stereo pair of sensory images. In the simplified version of this problem presented in (Becker and Hinton 1992), the goal is to learn that the visual input consists of randomly positioned dots on a 1-dimensional surface placed at one of two depths.

In our experiments, 4 blurred dots (Gaussian functions) were randomly positioned uniformly on the continuous interval $[0, 12]$ and the brightness of each dot (magnitude of the Gaussian) was drawn uniformly from $[0, 5]$. Next, either a left shift or a right shift was applied with equal probability to obtain a second activity pattern. Finally, two sensory images containing 12 real values each were obtained by dividing each interval into 12 pixels and assigning to each pixel the net activity within the pixel. Twelve examples from a training set of 1000 pairs of images obtained in this manner are shown in Fig. 2a, where the images are positioned so that the relative shift is evident. A noisy version of this data in which unit-variance Gaussian noise is added to each sensor is shown in Fig. 2e.

The stereo disparity problem is much more difficult than the bars problem, since there is more overlap between the underlying features. To see this, imagine a "multi-eyed" disparity problem in which there are as many sensory images as there are 1-dimensional sensors. We expect the depth inference to be easier in this case, since there is more evidence for each of the two possible directions of shift. Imagine stacking the sensory images on top of each other, so that each resulting square image will contain blurred diagonal bars that are oriented either up and to the right or up and to the left. Extracting disparity from this data is roughly equivalent to extracting bar orientation in the data from the previous section.

For each of the noisy and noise-free data sets we used 100 iterations of variational EM to train a three-layer NLGBN with 1 binary top-layer unit, 20 rectified middle-layer units, and 24 linear visible bottom-layer units. The resulting weights projecting from each of the 20 middle-layer units to the two sets of 12 pixels are shown in Fig. 2b and Fig. 2f. In both cases, the algorithm has extracted features that are spatially local and represent each of the two possible depths. Fig. 2c and Fig. 2g show the weights learned by variational EM in a network where all of the hidden units are binary. The log-probability bounds for the trained binary-rectified-linear NLGBNs are 1.0 nats and -43.7 nats, whereas the bounds for the trained binary-binary-linear NLGBNs are -37.8 nats and -46.1 nats. The Gibbs sampling method with rectified hidden units and using the same learning parameters as described in the previous section produced the weight patterns shown in Fig. 2d and Fig. 2h. For the noisy data, the features extracted by Gibbs sampling appear to be slightly cleaner than those extracted by variational EM.

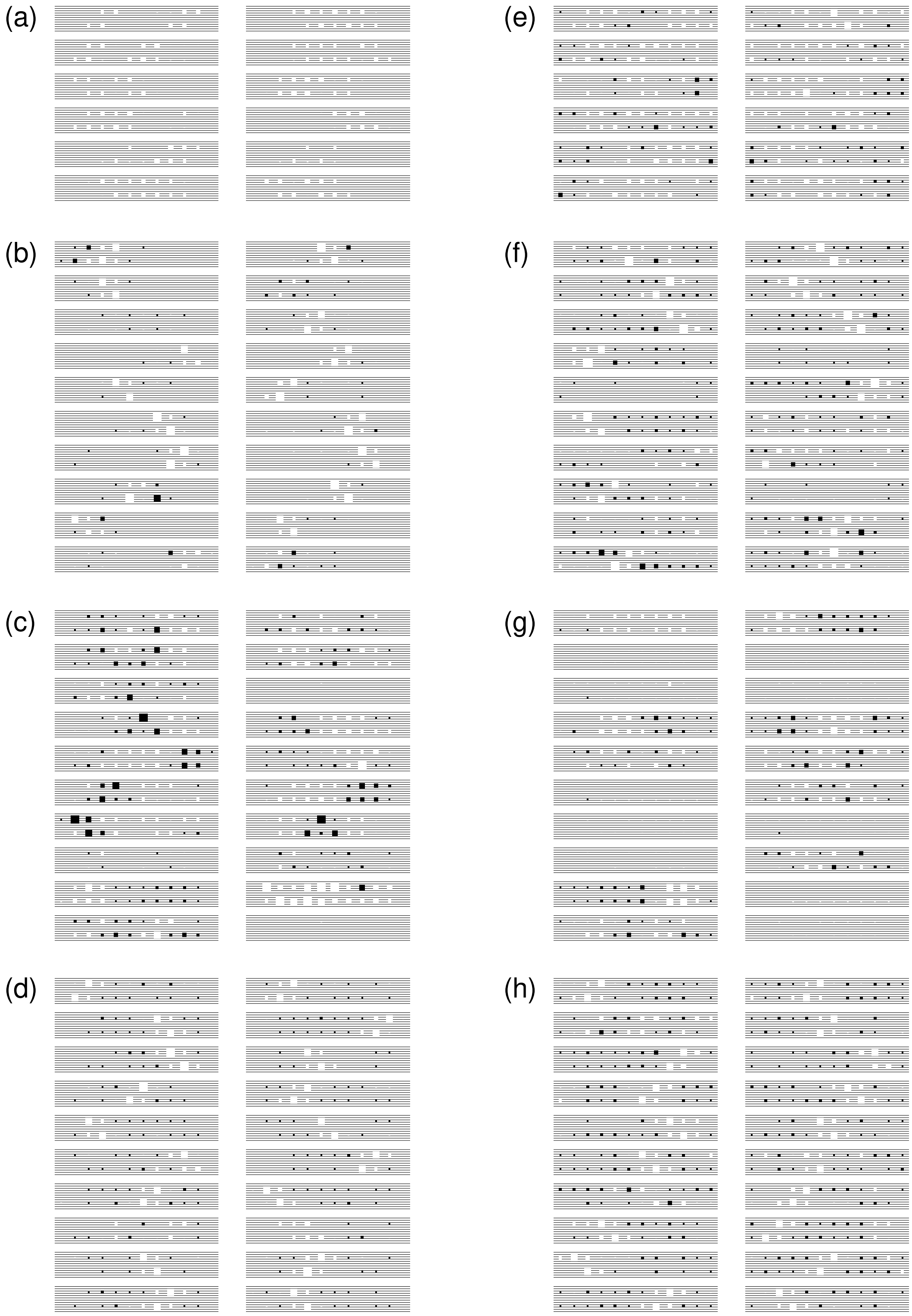

Figure 2: Learning in NLGBNs using the variational method and Gibbs sampling, for noise-free (a) - (d) and noisy (e) - (h) stereo disparity patterns. (a) and (e): Training examples. (b) and (f): Weights learned by the variational method with rectified units. (c) and (g): Weights learned by the variational method with binary units. (d) and (h): Weights learned by Gibbs sampling with rectified units.

4 Handwriting Recognition

Variational inference and learning in NLGBNs can be used to do real-valued pattern classification by training one NLGBN on each class of data. If C_i is the event that a pattern comes from class $i \in \{0, 1, \dots\}$, then the posterior class probabilities are given by Bayes' rule:

$$P(C_i|\{x_k\}_{k\in V}) = \frac{p(\{x_k\}_{k\in V}|C_i)P(C_i)}{\sum_j p(\{x_k\}_{k\in V}|C_j)P(C_j)}, \tag{9}$$

where $P(C_j)$ is the prior probability that a pattern comes from class j. The likelihood $p(\{x_k\}_{k\in V}|C_j)$ for class j is approximated by the value of the variational bound obtained from a generalized E-step.

In this section, we report the performances of several completely automated learning procedures on the problem of recognizing grey-level 8×8 images of handwritten digits from the CEDAR CDROM 1 database of ZIP codes (Hull 1994). The DELVE evaluation platform (Rasmussen et al. 1996) was used to obtain fair comparisons, including levels of statistical significance.

For each of three different sizes of training set, we empirically estimated the performances of k-nearest neighbors, with the neighborhood for each class of data determined using leave-one-out cross validation; mixture of diagonal (axis-aligned) Gaussians, with the number of Gaussians for each class determined using a validation set; factor analysis, with the number of factors for each class determined using a validation set; 1- and 2-hidden layer NLGBNs using rectified hidden units and linear visible units, with the number of hidden units determined using a validation set. For each of the latter four methods, the training set was first split according to class and then 1/3 of the data was set aside for validation. Models with different complexities (number of Gaussians or number of hidden units) were trained on the remaining 2/3 of the data using EM or generalized EM until convergence. The model that gave the highest validation set log-probability (or log-probability bound) was further trained to convergence on *all* of the data for the corresponding class. To prevent degenerate over-fitting of a pixel that happens to have the same value in all of the training cases, the variances for the visible units were not allowed to fall below 0.01 in the latter four methods. For each method, (9) was used to classify each test pattern.

To obtain robust estimates of the relative performances of these methods on the problem of handwritten digit recognition, we trained and tested each method multiple times using disjoint training set - test set pairs. The original data set of 11,000 8×8 images was partitioned into a set of 8000 images used for training and a set of 3000 images used for testing. For training set sizes of 1000 and 2000 patterns, 4 disjoint training set - test set pairs were extracted from the two partitions (each test set had 750 images). For the training set size of 4000 patterns, 2 disjoint training set - test set pairs were

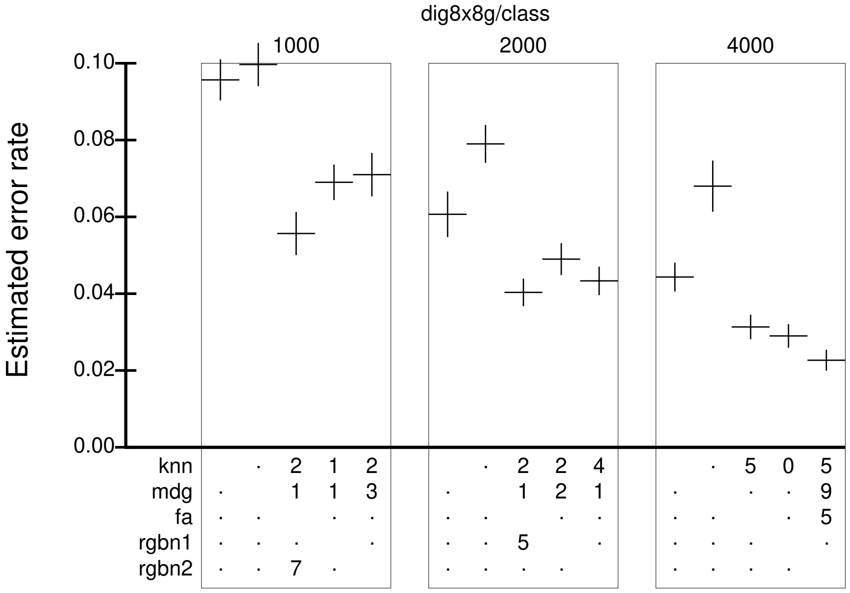

Figure 3: Estimated error rates on grey-level handwritten digit recognition using different sizes of training set (1000, 2000 and 4000 images) for the following methods: k-nearest neighbor, mixture of diagonal (axis-aligned) Gaussians, factor analysis, and rectified Gaussian belief networks with one and two hidden layers. For each size of training set, the error rates for the different methods are given in the above order. The numbers in the boxes are p-values (in percent) for a paired t-test on the null hypothesis that the corresponding two methods have identical performance (a dot indicates the p-value was above 9%).

extracted (each test set had 1500 images).

The results are shown in Fig. 3, where each horizontal bar gives an estimate of the expected error rate for a particular method using a particular training set size. The methods are ordered from left to right for each training set size as follows: k-nearest neighbors, mixture of Gaussians, factor analysis, 1-hidden layer NLGBN and 2-hidden layer NLGBN. Each vertical bar gives an estimate of the error (one standard deviation) in the corresponding estimate of the expected error rate. Integers in the boxes lying beneath the x-axis are p-values (in percent) for a paired t-test that compares the performances of the corresponding methods. Select a method from the list in the lower left-hand corner of the figure and scan from left to right. Whenever you see a number, that means another method has performed better than the method you selected, *with the given statistical significance*. A low p-value indicates the difference in the misclassification rates is significant. More precisely, the p-value is an estimate of the probability of obtaining a difference in performance that is equal to or greater than the observed difference, given that the true difference is zero (the null hypothesis).

On the largest training set size, the 2-hidden layer NLGBN performs better than k-nearest neighbors ($p = 5\%$), mixture of Gaussians ($p = 9\%$), factor analysis ($p = 5\%$) and the 1-hidden layer NLGBN ($p = 16\%$). The performance of the NLGBN with 1 hidden layer is fairly indistinguishable from the performance of factor analysis ($p = 25\%$). However, on average the NLGBN used only half as many hidden units as were used by factor analysis, indicating that the NLGBN provides a more compact representation of the input. It is also interesting that this more compact representation emerged despite the fact that the posterior distribution over the hidden units in the NLGBN was approximated by an axis-aligned Gaussian distribution, whereas in factor analysis the exact full-covariance posterior distribution over the hidden units is used.

5 Conclusions

Results on visual feature extraction show that the variational technique can extract perceptually significant continuous nonlinear latent structure. In contrast with networks with continuous hidden variables, networks with binary hidden variables do not extract spatially local features from the data. Similarly, linear methods like factor analysis and independent component analysis fail to extract spatially local features (Zoubin Ghahramani, personal communication). Results also show that the variational method presented in this paper is a viable alternative to Gibbs sampling in stochastic neural networks with rectified hidden variables.

Advantages of the variational method over Gibbs sampling include the absence of a learning rate and the ability to compute the log-probability bound very efficiently. The latter is particularly useful for pattern classifiers that train one network on each class of data and then classify a novel pattern by picking the network that gives the highest estimate of the log-probability (Frey 1998). Results show that for handwritten digit recognition, there is a regime of training set size in which NLGBNs perform better than k-nearest neighbors, mixture of Gaussians and the linear factor analysis method.

The variational method may be made more powerful by making each distribution $q(x_i)$ in (5) a mixture of Gaussians, by making the entire distribution $q(\cdot)$ a mixture of product-form distributions (Jaakkola and Jordan 1998), or by grouping together small numbers of hidden variables over which full-covariance Gaussians are fit during variational inference.

We considered three types of continuous nonlinear unit in this paper: binary, rectified, and sigmoidal. However, the variational method can easily be extended to other types of units (such as "twinned" units (Hinton, Sallans and Ghahramani 1998)) as long as the output mean function $M(\mu, \sigma)$ and the output variance function $V(\mu, \sigma)$ can be computed. To perform a gradient-based E-step, the gradients of these functions with respect to their arguments are also needed.

Although we have not focussed our attention on implementations of the variational algorithm that are suited to biology, we believe that with some modifications they can be made so. The inference algorithm uses the log-probability bound derivatives given in (12), and these are computed from simple differences passed locally in the network. A partial M-Step can be used for on-line learning, in which case the derivative of the bound for just the current input pattern is followed. This derivative can be followed by applying a delta-type rule based on locally computed differences.

Acknowledgments

We thank Peter Dayan, Zoubin Ghahramani and Tommi Jaakkola for helpful discussions. We also appreciate the useful feedback provided on this manuscript by Michael Jordan, an anonymous reviewer and Karla Miller. The Gibbs sampling experiments were performed using software developed by Zoubin Ghahramani (see `http://www.cs.utoronto.ca/~zoubin`). This research was funded by grants from the Arnold and Mabel Beckman Foundation, the Natural Science and Engineering Research Council of Canada and the Information Technology Research Center of Ontario.

A The E-Step

Here, we show how to compute F and its derivatives with respect to the variational parameters. For the current set of variational parameters (including the ones fixed by the current input pattern), we first compute for each unit the current values of the mean output $m_i \leftarrow M_i(\mu_i, \sigma_i)$, the output variance $v_i \leftarrow V_i(\mu_i, \sigma_i)$, and the mean *net input*

$$n_i \leftarrow \sum_{j \in A_i} w_{ij} m_j. \tag{10}$$

Then, the bound on the log-probability of the input pattern is computed from

$$F \leftarrow -\sum_{i=1}^{N} \frac{1}{2\psi_i^2} \left[(\mu_i - n_i)^2 + \textstyle\sum_{j \in A_i} w_{ij}^2 v_j \right]$$
$$+ \sum_{i \in H} \frac{1}{2} \left(1 + \log 2\pi\sigma_i^2 - \frac{\sigma_i^2}{\psi_i^2} \right) - \sum_{i=1}^{N} \frac{1}{2} \log 2\pi\psi_i^2. \tag{11}$$

To perform a gradient-based optimization in the E-Step, the derivatives of the bound with respect to μ_j and $\log \sigma_j^2$ for $j \in H$ are computed as follows:

$$\frac{\partial F}{\partial \mu_j} \leftarrow \frac{n_j - \mu_j}{\psi_j^2} - \frac{\partial M_j(\mu_j, \sigma_j)}{\partial \mu_j} \sum_{i \in C_j} \frac{w_{ij}(n_i - \mu_i)}{\psi_i^2} - \frac{\partial V_j(\mu_j, \sigma_j)}{\partial \mu_j} \sum_{i \in C_j} \frac{w_{ij}^2}{2\psi_i^2},$$
$$\frac{\partial F}{\partial \log \sigma_j^2} \leftarrow -\frac{\partial V_j(\mu_j, \sigma_j)}{\partial \log \sigma_j^2} \sum_{i \in C_j} \frac{w_{ij}^2}{2\psi_i^2} - \frac{\partial M_j(\mu_j, \sigma_j)}{\partial \log \sigma_j^2} \sum_{i \in C_j} \frac{w_{ij}(n_i - \mu_i)}{\psi_i^2} - \frac{\sigma_j^2}{2\psi_j^2} + \frac{1}{2}, \tag{12}$$

where C_j is the set of indices for the children of unit j. App. C gives expressions for the derivatives of the nonlinear functions for binary units, rectified units and sigmoidal units.

An E-Step produces one set of variational parameters $\mu_i^{(t)}$, $\sigma_i^{(t)}$, $i = 1, \ldots, N$ and the corresponding $m_i^{(t)}$, $v_i^{(t)}$, and $n_i^{(t)}$, $i = 1, \ldots, N$ for each training pattern, $t = 1, \ldots, T$. These are used to initialize the next E-Step.

B The M-Step

In the log-probability bound in (8), the model variances do not influence the optimal weights. So, in the M-Step, we first maximize the total bound with respect to the weights. Since the bound is quadratic in the weights, we use singular value decomposition to solve for them exactly. In fact, the weights associated with the input to each variable are decoupled from the other weights in the network. That is, the value of w_{ij} does not affect the optimal value of w_{kl} if $i \neq k$. Consequently, solving for the optimal weights is a matter of solving N linear systems, where system i, $i = 1, \ldots, N$, has dimensionality equal to the number of parents for unit i, and once solved gives the weights on the incoming connections to unit i.

Consider the input means $\mu_i^{(t)}$, output means $m_i^{(t)}$, input variances $\sigma_i^{(t)}$, output variances $v_i^{(t)}$, and mean net inputs $n_i^{(t)}$, $i = 1, \ldots, N$, that are computed for training pattern t in the E-Step. It is not necessary to store all of these sets for all T training patterns. However, they are used to compute the following sufficient statistics:

$$a_{jk} \leftarrow \frac{1}{T} \sum_t m_j^{(t)} m_k^{(t)}, \quad b_j \leftarrow \frac{1}{T} \sum_t v_j^{(t)}, \quad c_{ij} \leftarrow \frac{1}{T} \sum_t \mu_i^{(t)} m_j^{(t)},$$
$$d_j \leftarrow \frac{1}{T} \sum_t (n_j^{(t)} - \mu_j^{(t)})^2, \quad e_j \leftarrow \frac{1}{T} \sum_t {\sigma_j^{(t)}}^2, \tag{13}$$

for $j = 0, \ldots, N$, $k = 0, \ldots, N$, and $i = j+1, \ldots, N$. These can be accumulated while scanning through the training set during the E-Step.

Once the sufficient statistics have been computed, we first solve for the weights. The system of equations for the weights associated with the input to unit i is

$$\sum_{k \in A_i} a_{jk} w_{ik} + b_j w_{ij} = c_{ij}, \quad j \in A_i, \tag{14}$$

where i is fixed in this set of equations. We use singular value decomposition to solve for each set of weights. In fact, the system of equations for unit i has dimensionality equal to the number of parents for unit i (including its bias). Finally, the model variances are computed from

$$\psi_j^2 \leftarrow d_j + e_j + \sum_{k \in A_j} w_{jk}^2 b_k. \tag{15}$$

C $M(\mu, \sigma)$, $V(\mu, \sigma)$ and their derivatives for interesting nonlinear functions

In this appendix, we give the output means and variances for some useful types of units, including linear units, binary units, rectified units, and sigmoidal units.

C.1 Linear units

Although this paper is about how to deal with nonlinear units, it is often useful to include some units (*e.g.*, visible units) that are linear:

$$f(x) = x. \tag{16}$$

For this unit, the output mean and variance are

$$M(\mu, \sigma) = \mu, \qquad V(\mu, \sigma) = \sigma^2. \tag{17}$$

The derivatives are

$$\begin{aligned} \frac{\partial M(\mu,\sigma)}{\partial \mu} &= 1, \qquad \frac{\partial M(\mu,\sigma)}{\partial \log \sigma^2} = 0, \\ \frac{\partial V(\mu,\sigma)}{\partial \mu} &= 0, \qquad \frac{\partial V(\mu,\sigma)}{\partial \log \sigma^2} = \sigma^2. \end{aligned} \tag{18}$$

C.2 Binary units

To obtain a stochastic binary unit, take

$$f(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases} \tag{19}$$

For this unit, the output mean and variance are

$$M(\mu, \sigma) = \Phi\Big(\frac{\mu}{\sigma}\Big), \qquad V(\mu, \sigma) = \Phi\Big(\frac{\mu}{\sigma}\Big)\Big[1 - \Phi\Big(\frac{\mu}{\sigma}\Big)\Big], \tag{20}$$

where $\Phi(\cdot)$ is the cumulative Gaussian function:

$$\Phi(y) = \int_{\alpha=-\infty}^{y} \phi(\alpha) d\alpha. \tag{21}$$

The derivatives are

$$\begin{aligned}
\frac{\partial M(\mu,\sigma)}{\partial \mu} &= \frac{1}{\sigma}\phi\Big(\frac{\mu}{\sigma}\Big), \qquad \frac{\partial M(\mu,\sigma)}{\partial \log \sigma^2} = -\frac{\mu}{2\sigma}\phi\Big(\frac{\mu}{\sigma}\Big), \\
\frac{\partial V(\mu,\sigma)}{\partial \mu} &= \frac{1}{\sigma}\phi\Big(\frac{\mu}{\sigma}\Big)\Big[1 - 2\Phi\Big(\frac{\mu}{\sigma}\Big)\Big], \\
\frac{\partial V(\mu,\sigma)}{\partial \log \sigma^2} &= -\frac{\mu}{2\sigma}\phi\Big(\frac{\mu}{\sigma}\Big)\Big[1 - 2\Phi\Big(\frac{\mu}{\sigma}\Big)\Big].
\end{aligned} \tag{22}$$

C.3 Rectified units

A rectified unit is linear if its input exceeds 0, and outputs 0 otherwise:

$$f(x) = \begin{cases} 0 & \text{if } x < 0, \\ x & \text{if } x \geq 0. \end{cases} \tag{23}$$

For this unit, the output mean and variance are

$$\begin{aligned}
M(\mu,\sigma) &= \mu\Phi\Big(\frac{\mu}{\sigma}\Big) + \sigma\phi\Big(\frac{\mu}{\sigma}\Big), \\
V(\mu,\sigma) &= (\mu^2 + \sigma^2)\Phi\Big(\frac{\mu}{\sigma}\Big) + \mu\sigma\phi\Big(\frac{\mu}{\sigma}\Big) - M(\mu,\sigma)^2.
\end{aligned} \tag{24}$$

The derivatives are

$$\begin{aligned}
\frac{\partial M(\mu,\sigma)}{\partial \mu} &= \Phi\Big(\frac{\mu}{\sigma}\Big), \qquad \frac{\partial M(\mu,\sigma)}{\partial \log \sigma^2} = \frac{\sigma}{2}\phi\Big(\frac{\mu}{\sigma}\Big), \\
\frac{\partial V(\mu,\sigma)}{\partial \mu} &= 2\mu\Phi\Big(\frac{\mu}{\sigma}\Big) + 2\sigma\phi\Big(\frac{\mu}{\sigma}\Big) - 2M(\mu,\sigma)\Phi\Big(\frac{\mu}{\sigma}\Big), \\
\frac{\partial V(\mu,\sigma)}{\partial \log \sigma^2} &= \sigma^2\Phi\Big(\frac{\mu}{\sigma}\Big) - \sigma M(\mu,\sigma)\Phi\Big(\frac{\mu}{\sigma}\Big).
\end{aligned} \tag{25}$$

C.4 Sigmoidal units

The cumulative Gaussian squashing function,

$$f(x) = \Phi(x), \tag{26}$$

leads to closed-form expressions for the output mean and its derivatives. We have not found a closed-form expression for the output variance, but it can be approximately bounded by a new function $V'(\mu, \sigma)$, giving an upper bound on the free energy. The output mean and variance bound are

$$M(\mu, \sigma) = \Phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big),$$

$$V(\mu, \sigma) \leq V'(\mu, \sigma) = \Phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big[1 - \Phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big]\frac{\sigma^2}{\sigma^2 + \pi/2}. \tag{27}$$

The derivatives are

$$\frac{\partial M(\mu, \sigma)}{\partial \mu} = \frac{1}{\sqrt{1+\sigma^2}}\phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big),$$

$$\frac{\partial M(\mu, \sigma)}{\partial \log \sigma^2} = -\frac{\mu\sigma^2}{2(1+\sigma^2)^{3/2}}\phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big),$$

$$\frac{\partial V'(\mu, \sigma)}{\partial \mu} = \frac{\sigma^2}{(\sigma^2 + \pi/2)\sqrt{1+\sigma^2}}\phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big[1 - 2\Phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big],$$

$$\begin{aligned}\frac{\partial V'(\mu, \sigma)}{\partial \log \sigma^2} = \frac{\sigma^2}{\sigma^2 + \pi/2}\Big\{&\frac{\pi/2}{\sigma^2 + \pi/2}\Phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big[1 - \Phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big] \\ &-\frac{\mu\sigma^2}{2(1+\sigma^2)^{3/2}}\phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big[1 - 2\Phi\Big(\frac{\mu}{\sqrt{1+\sigma^2}}\Big)\Big]\Big\}.\end{aligned} \tag{28}$$

References

Amari, S.-I. 1985. *Differential-Geometrical Methods in Statistics, Lecture Notes in Statistics vol. 28*. Springer, New York NY.

Amari, S.-I., Cichocki, A., and Yang, H. 1996. A new learning algorithm for blind signal separation. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge MA.

Becker, S. and Hinton, G. E. 1992. A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355:161–163.

Bell, A. J. and Sejnowski, T. J. 1995. An information maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159.

Comon, P., Jutten, C., and Herault, J. 1991. Blind separation of sources. *Signal Processing*, 24:11–20.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. 1995. The Helmholtz machine. *Neural Computation*, 7:889–904.

Dayan, P. and Zemel, R. S. 1995. Competition and multiple cause models. *Neural Computation*, 7:565–579.

Everitt, B. S. 1984. *An Introduction to Latent Variable Models*. Chapman and Hall, New York NY.

Frey, B. J. 1997a. Continuous sigmoidal belief networks trained using slice sampling. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*. MIT Press, Cambridge MA. Available at `http://www.cs.utoronto.ca/~frey`.

Frey, B. J. 1997b. Variational inference for continuous sigmoidal Bayesian networks. In *Sixth International Workshop on Artificial Intelligence and Statistics*.

Frey, B. J. 1998. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge MA. See `http:// mitpress.mit.edu/ book-home.tcl? isbn=026206202X`.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. 1995. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1158–1161.

Hinton, G. E. and Ghahramani, Z. 1997. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London B*, 352:1177–1190.

Hinton, G. E., Sallans, B., and Ghahramani, Z. 1998. A hierarchical community of experts. In Jordan, M. I., editor, *Learning and Inference in Graphical Models*. Kluwer Academic Publishers, Norwell MA.

Hinton, G. E. and Sejnowski, T. J. 1986. Learning and relearning in Boltzmann machines. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume I, pages 282–317. MIT Press, Cambridge MA.

Hull, J. J. 1994. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554.

Jaakkola, T., Saul, L. K., and Jordan, M. I. 1996. Fast learning by bounding likelihoods in sigmoid type belief networks. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*. MIT Press.

Jaakkola, T. S. and Jordan, M. I. 1998. Approximating posteriors via mixture models. In Jordan, M. I., editor, *Learning and Inference in Graphical Models*. Kluwer Academic Publishers, Norwell MA.

Lewicki, M. S. and Sejnowski, T. J. 1998. Learning nonlinear overcomplete representations for efficient coding. In Jordan, M. I., Kearns, M. I., and Solla, S. A., editors, *Advances in Neural Information Processing Systems 10*. MIT Press, Cambridge MA.

MacKay, D. J. C. 1997. Maximum likelihood and covariant algorithms for independent component analysis. Unpublished manuscript available at `http://wol.ra.phy.cam.ac.uk/mackay`.

Neal, R. M. 1992. Connectionist learning of belief networks. *Artificial Intelligence*, 56:71–113.

Neal, R. M. and Hinton, G. E. 1993. A new view of the EM algorithm that justifies incremental and other variants. Unpublished manuscript available over the internet by ftp at `ftp://ftp.cs.utoronto.ca/pub/radford/em.ps.Z`.

Olshausen, B. A. and Field, D. J. 1996. Emergence of simple-cell receptive-field properties by learning a sparse code for natural images. *Nature*, 381:607–609.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo CA.

Rasmussen, C. E., Neal, R. M., Hinton, G. E., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., and Tibshirani, R. 1996. *The DELVE Manual*. University of Toronto, Toronto Canada. `http://www.cs.utoronto.ca/~delve`.

Rubin, D. and Thayer, D. 1982. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76.

Saul, L. K., Jaakkola, T., and Jordan, M. I. 1996. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76.